AMENDMENTS TO THE CLAIMS

Claim 1 (previously presented): In a system for executing write and read data commands, the system having a shared read/write buffer pool of blocks for temporarily storing write data to be sent to a peer device and read data received from the peer device, an apparatus for managing read and write data congestion in the buffer pool, the apparatus comprising:

a processor programmed for preventing an initiation of a new write data command until pending read data requests have been processed enough to free up sufficient blocks in the buffer pool to accommodate the data of the new write data command.

Claim 2 (previously presented): The apparatus as recited in claim 1, the processor further programmed for:

determining a number of blocks that will be required to store the read or write data for the pending read data requests and the new write data command;

determining a number of free blocks in the buffer pool; and

throttling the new write data command if the number of free blocks is insufficient to store the read ~~or~~data for the pending read data requests and the write data for the new write data command.

Claim 3 (original): The apparatus as recited in claim 2, the system comprising a receive list memory which contains descriptor pointers to free blocks and blocks filled with read data, and a free list memory which contains descriptor pointers to free blocks not referenced in the receive list memory, the processor further programmed for determining the number of free blocks in the buffer pool by:

summing the number of free blocks in the receive list memory and the free list memory.

Claim 4 (previously presented): The apparatus as recited in claim 2, the processor further programmed for initiating the new write data command if the number of free blocks is

sufficient to store the read data for the pending read data requests and the write data for the new write data command.

Claim 5 (previously presented): The apparatus as recited in claim 2, the system comprising a receive list memory which contains descriptor pointers to free blocks and blocks filled with read data, and a free list memory which contains descriptor pointers to free blocks not referenced in the receive list memory, the processor further programmed for determining the number of free blocks in the buffer pool by:

summing the number of free blocks in the receive list memory and the free list memory; and

subtracting from the sum a number of blocks estimated for storing incoming read data for any pending read data requests.

Claim 6 (previously presented): The apparatus as recited in claim 5, the processor further programmed for initiating the new read or write data command if the number of free blocks is sufficient to store the read data for the pending read data requests and the write data for the new write data command.

Claim 7 (previously presented): The apparatus as recited in claim 2, the processor further programmed for:

storing the throttled new write data command and any subsequent new read or write data commands into a first-in-first-out (FIFO) read/write command request queue;

processing pending read data requests to completion to free up blocks in the buffer pool; and

executing a next read or write data command from the read/write command request queue if the number of free blocks becomes sufficient to store the read or write data for the next read or write data command.

Claim 8 (original): A host bus adapter (HBA) comprising the apparatus of claim 1, the HBA for implementing upper layer protocols (ULPs).

Claim 9 (original): The HBA of claim 8, further comprising an Internet Small Computer System Interface (iSCSI) controller circuit.

Claim 10 (original): A host computer comprising the HBA of claim 9.

Claim 11 (original): A storage area network (SAN) comprising the host computer of claim 10, wherein an iSCSI network is coupled to the iSCSI controller circuit and one or more storage devices are coupled to the iSCSI network.

Claim 12 (previously presented): A computer program for avoiding read and write data congestion in a system for executing write and read data commands and having a shared read/write buffer pool of blocks for temporarily storing write data to be sent to a peer device and read data received from the peer device, the computer program being stored on a machine readable medium and executable to perform acts comprising:

preventing an initiation of a new write data command until pending read data requests have been processed enough to free up sufficient blocks in the buffer pool to accommodate the data of the new write data command.

Claim 13 (previously presented): The computer program as recited in claim 12, further executable to perform acts comprising:

determining a number of blocks that will be required to store the read or write data for the pending read data requests and the new write data command;

determining a number of free blocks in the buffer pool; and

throttling the new write data command if the number of free blocks is insufficient to store the read data for the pending read data requests and the write data for the new read or write data command.

Claim 14 (original): The computer program as recited in claim 13, the system comprising a receive list memory which contains descriptor pointers to free blocks and blocks filled with read data, and a free list memory which contains descriptor pointers to free blocks not

referenced in the receive list memory, the computer program further executable to perform acts comprising determining the number of free blocks in the buffer pool by:

summing the number of free blocks in the receive list memory and the free list memory.

Claim 15 (previously presented): The computer program as recited in claim 13, further executable to perform acts comprising initiating the new write data command if the number of free blocks is sufficient to store the read data for the pending read data requests and the write data for the new write data command.

Claim 16 (previously presented): The computer program as recited in claim 13, the system comprising a receive list memory which contains descriptor pointers to free blocks and blocks filled with read data, and a free list memory which contains descriptor pointers to free blocks not referenced in the receive list memory, the computer program further executable to perform acts comprising determining the number of free blocks in the buffer pool by:

summing the number of free blocks in the receive list memory and the free list memory; and

subtracting from the sum a number of blocks estimated for storing incoming read data for any pending read data requests.

Claim 17 (previously presented): The computer program as recited in claim 16, further executable to perform acts comprising initiating the new read or write data command if the number of free blocks is sufficient to store the read data for the pending read data requests and the write data for the new write data command.

Claim 18 (previously presented): The computer program as recited in claim 13, further executable to perform acts comprising:

storing the throttled new write data command and any subsequent new read or write data commands into a first-in-first-out (FIFO) read/write command request queue;

processing pending read data requests to completion to free up blocks in the buffer pool; and

executing a next read or write data command from the read/write command request queue if the number of free blocks becomes sufficient to store the read or write data for the next read or write data command.

Claim 19 (original): A host bus adapter (HBA) comprising the computer program of claim 12, the HBA for implementing upper layer protocols (ULPs).

Claim 20 (original): The HBA of claim 19, further comprising an Internet Small Computer System Interface (iSCSI) controller circuit.

Claim 21 (original): A host computer comprising the HBA of claim 20.

Claim 22 (original): A storage area network (SAN) comprising the host computer of claim 21, wherein an iSCSI network is coupled to the iSCSI controller circuit and one or more storage devices are coupled to the iSCSI network.

Claim 23 (previously presented): A method for avoiding read and write data congestion in a system for executing write and read data commands and having a shared read/write buffer pool of blocks for temporarily storing write data to be sent to a peer device and read data received from the peer device, the method comprising:

preventing an initiation of a new write data command until pending read data requests have been processed enough to free up sufficient blocks in the buffer pool to accommodate the data of the new write data command.

Claim 24 (previously presented): The method as recited in claim 23, further comprising:

determining a number of blocks that will be required to store the read or write data for the pending read data requests and the new write data command;

determining a number of free blocks in the buffer pool; and

throttling the new write data command if the number of free blocks is insufficient to store the read data for the pending read data requests and the write data for the new write data command.

Claim 25 (original): The method as recited in claim 24, the system comprising a receive list memory which contains descriptor pointers to free blocks and blocks filled with read data, and a free list memory which contains descriptor pointers to free blocks not referenced in the receive list memory, the step of determining the number of free blocks in the buffer pool further comprising:

summing the number of free blocks in the receive list memory and the free list memory.

Claim 26 (previously presented): The method as recited in claim 24, further comprising initiating the new write data command if the number of free blocks is sufficient to store the read data for the pending read data requests and the write data for the new write data command.

Claim 27 (previously presented): The method as recited in claim 24, the system comprising a receive list memory which contains descriptor pointers to free blocks and blocks filled with read data, and a free list memory which contains descriptor pointers to free blocks not referenced in the receive list memory, the step of determining the number of free blocks in the buffer pool further comprising:

summing the number of free blocks in the receive list memory and the free list memory; and

subtracting from the sum ~~to~~ a number of blocks estimated for storing incoming read data for any pending read data requests.

Claim 28 (previously presented): The method as recited in claim 27, further comprising initiating the new read or write data command if the number of free blocks is sufficient to store the read data for the pending read data requests and the write data for the new write data command.

Claim 29 (previously presented): The method as recited in claim 24, further comprising:

storing the throttled new write data command and any subsequent new read or write data commands into a first-in-first-out (FIFO) read/write command request queue;

processing pending read data requests to completion to free up blocks in the buffer pool; and

executing a next read or write data command from the read/write command request queue if the number of free blocks becomes sufficient to store the read or write data for the next read or write data command.